



4 Access und VBA

Ziele dieses Kapitels

- ▶ Sie vertiefen Ihr VBA-Wissen, indem Sie weitere Beispiele, diesmal von ACCESS realisieren.
- ▶ Sie lernen die Verbindung zu anderen Office-Programmen, indem Sie von ACCESS Excel-VBA-Befehle aufrufen.
- ▶ Sie erfahren etwas über die beiden Datenzugriffsmethoden DAO und ADO.

4.1 Aufgabenstellung

Im Beispiel „Jahresplan“ haben Sie die Entwicklung eines Jahreskalenders von EXCEL kennengelernt. Der dort installierte Kalender kann auch in Verbindung mit anderen Programmen zum Einsatz kommen. Stellen Sie sich zum Beispiel eine Seminarverwaltung vor. Der Mitarbeiter, der alles koordiniert, was zum Seminarbetrieb erforderlich ist, möchte den Jahresplan nutzen, um eine Raum- und Trainerplanung vorzunehmen.

Schön wäre es, wenn in der Seminarverwaltung ein Auswahlformular existierte, mit dem, nach der Auswahl des Raumes oder des Trainers, ein Jahresplan aufbereitet wird, und zwar unter Berücksichtigung der in der Datenbank gespeicherten Informationen.



4.1.1 Aufbau der Beispieldatenbank

Die nachfolgend abgebildete Struktur zeigt, wie eine Beispieldatenbank in ACCESS aussehen könnte, die wir für unsere Realisierung benötigen. Bevor wir näher darauf eingehen, sollten Sie sich vor allem die Beziehungen näher anschauen.

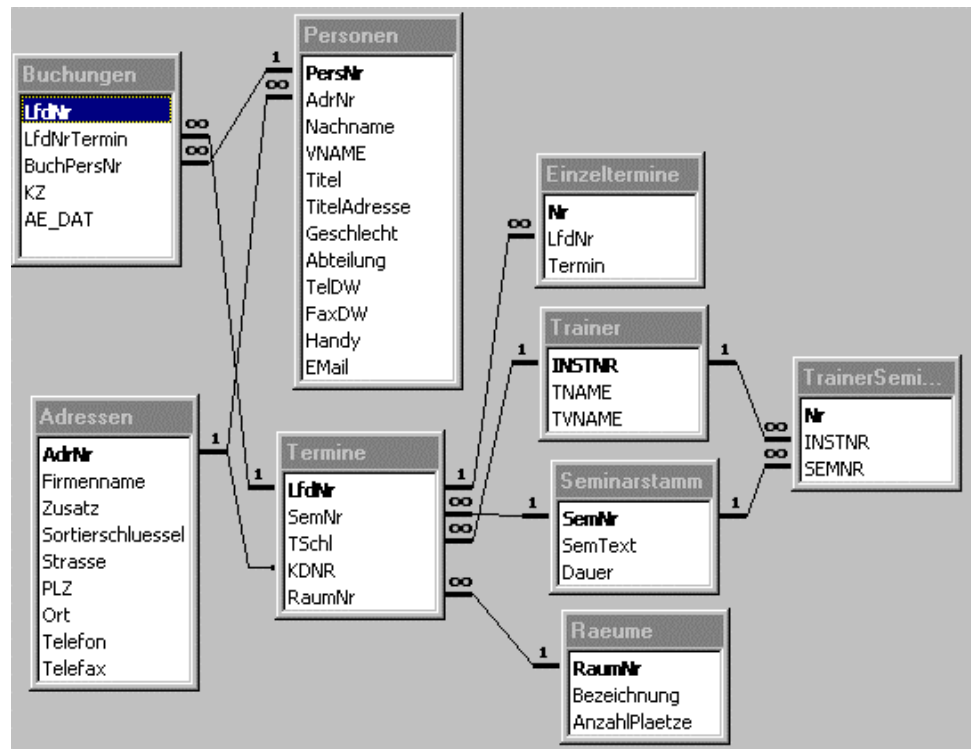


Abb. 4.1: Datenbankstruktur der Beispiel-Datenbank

Sie können erkennen, dass die einzelnen Termine für die Planung in der Tabelle **Einzeltermine** gespeichert sind. Um zu wissen, in welchem Raum das Seminar stattfinden, brauchen Sie noch die Tabelle **Termine**. Von dort kann die endgültige Beziehung zur Tabelle **Raeume** aufgebaut werden, in der die Bezeichnung des betreffenden Raums gespeichert ist.



4.1.2 Formulare

Formulare zur Eingabe und Änderung von Daten sind zwar wichtig, gehören jedoch nicht zum Themenkreis dieser Unterlage. Trotzdem müssen Sie zunächst ein Formular einrichten, mit dem Sie den Raum auswählen können, für den die Planung vorgesehen ist. Außerdem muss dieses Formular eine Schaltfläche enthalten, mit der die Aufbereitung gestartet wird.

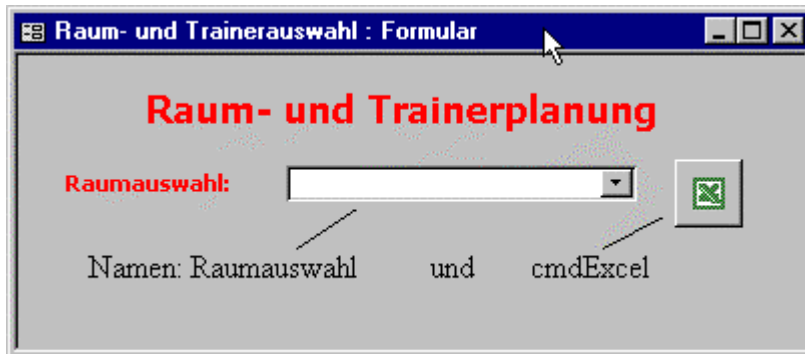


Abb. 4.2: Auswahlformular für die Raumplanung

Die benötigten Tabellen und das Formular finden Sie fertig vorbereitet in der Datenbank SEMI_01.MDB.

4.2 Realisierung

Für die Realisierung unserer Aufgabe werden wir wieder Teilschritte bilden. Das macht das Ganze übersichtlicher und damit einfacher. Folgende Einzelschritte sind nötig:

- ▶ 1. Teilaufgabe:
Da wir auf Excel-Befehle zurückgreifen, muss gewährleistet sein, dass eine Excel-Task geöffnet ist.
- ▶ 2. Teilaufgabe:
Die bisher von Hand aufgerufene Vorlage für die Jahresplanung muss jetzt von ACCESS gestartet werden.
- ▶ 3. Teilaufgabe:
Die für die Raumplanung benötigten Termine müssen von ACCESS bereitgestellt werden. Mit einem ADO-Zugriff werden Sie die Informationen der Jahresplanung hinzufügen.
- ▶ 4. Teilaufgabe:
Diese Teilaufgabe wird im Übungsteil realisiert. Sie enthält die Erweiterung um die Trainerplanung.



4.2.1 Excel von Access aus starten

Sie können innerhalb von ACCESS auf den Befehlsschatz von EXCEL zurückgreifen. Voraussetzung ist, dass EXCEL geöffnet ist und Sie einen Verweis zur Excel-Objektbibliothek vorgenommen haben.

Excel Bibliotheks- verweis



Sie aktivieren den Verweis zur Excel-Objektbibliothek.

1. Aktivieren Sie die Ereignisprozedur **Beim Klicken** der Schaltfläche **cmdExcel**.

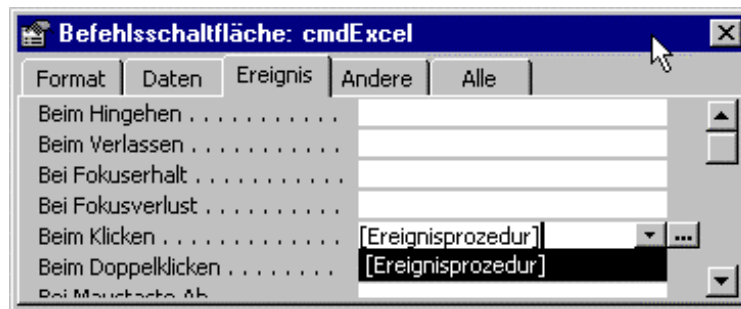


Abb. 4.3: Ereignisprozedur aktivieren

2. Aktivieren Sie den Verweis zur Excel-Objektbibliothek mit Hilfe des Befehls Extras → Verweise... → Microsoft Excel 9.0 Object Library .

Sie können jetzt auf den kompletten Befehlsvorrat dieser Bibliothek zurückgreifen.

Anschließend müssen Sie dafür Sorge tragen, dass EXCEL geöffnet ist. Dazu brauchen Sie eine Objektvariable, deren Zuweisung zunächst mit der Funktion **GetObject** erfolgt. Falls EXCEL bereits geöffnet ist, ist damit die Teilaufgabe bereits erledigt. Wenn nicht – Sie prüfen dies mit der Fehlerbehandlungsroutine – starten Sie EXCEL mit der **CreateObject**-Funktion. Die Anwendung muss dann noch sichtbar gemacht werden (Eigenschaft **visible**).

EXCEL starten



Sie sorgen dafür, dass EXCEL geöffnet ist.

1. Deklarieren Sie eine Objektvariable vom Typ **Application**.
Dim oApp As Excel.Application
2. Aktivieren Sie die Fehlerbehandlung
On Error Resume Next
3. Versuchen Sie einen Verweis zu EXCEL
Set oApp = GetObject(, "Excel.Application")



4. Falls ein Fehler aufgetreten ist – EXCEL ist dann nicht geöffnet –, löschen Sie das Err.Objekt und öffnen EXCEL mit der CreateObjekt-Funktion:
If Err.Number > 0 Then
Err.Clear
Set oApp = CreateObject("Excel.Application")
5. Falls erneut ein Fehler auftritt, ist EXCEL nicht zu öffnen, z.B. wenn EXCEL nicht installiert ist. Verlassen Sie die Prozedur mit einer entsprechenden Fehlermeldung.
6. Machen Sie Ihre Excel-Anwendung sichtbar
oApp.Visible = True
7. Setzen Sie die UserControl-Eigenschaft noch auf True, weil ansonsten EXCEL nach dem letzten Programmschritt wieder geschlossen wird.
oApp.UserControl = True
8. Falls gewünscht, sorgen Sie dafür, dass ACCESS anschließend wieder die aktuelle Anwendung ist.
AppActivate "Microsoft Access"
9. Speichern und testen Sie Ihre Anwendung.

Sie finden die komplette Lösung dieses Teilschritts wieder im Anhang Lösungen und in der Beispieldatenbank SEMI_02.MDB.

4.2.2 Excel-Bedienung von Access aus

Nachdem Sie einen Verweis zu EXCEL vorgenommen haben, können Sie alle Excel-VBA-Anweisungen auch von ACCESS aus aufrufen. Ein kleines Beispiel hierzu.

Sie erinnern sich, dass Sie mit dem **Range**-Objekt eine Zelle oder einen Zellbereich ansprechen konnten. Würden Sie direkt in EXCEL in die Zelle A1 die Jahreszahl 1999 schreiben wollen, lautete der Befehl:

```
Range("A1") = 1999
```

Damit VBA in ACCESS weiß, dass es sich um einen Excel-VBA-Befehl handelt, müssen Sie der Zuweisung diese Information mitliefern, indem Sie die Objektvariable, die Sie mit **GetObject** bzw. **CreateObject** gefüllt haben, der Anweisung vorweg stellen.

```
oApp.Range("A1") = 1999
```

Falls die genauen Befehle unbekannt sein sollten, hilft Ihnen der Makrorecorder bei der Befehlsermittlung.



Aufruf Jahresplan



Sie lassen den Jahresplan von ACCESS generieren.

1. Um die Befehle festzustellen, richten Sie in EXCEL zunächst eine neue, leere Arbeitsmappe ein und aktivieren den Makrorecorder mit dem Befehl: Extras → Makro → Aufzeichnen.... Das Dialogfeld bestätigen Sie ohne Änderung mit .
2. Rufen Sie die benötigten Befehle jetzt manuell auf. Befehle: Datei → Neu → Vorlage Jahresplan auswählen und .
3. Befolgen Sie die Anweisungen Ihres Excel-Makros.
4. Beenden Sie die Makroaufzeichnung:



Abb. 4.4: Ende der Makroaufzeichnung

5. Kontrollieren Sie das aufgezeichnete Makro mit dem Befehl: Extras → Makro → Makros..., wählen Sie das gerade aufgezeichnete Makro aus und betätigen die Schaltfläche .
6. Kopieren Sie den Befehl in die Zwischenablage und fügen ihn am Ende Ihrer Access-Ereignisprozedur ein.
7. Schreiben Sie noch den Namen der Objektvariablen vor die Anweisung, gefolgt von einem Punkt:
oApp.Workbooks.Add...
8. Da der Pfad mit aufgezeichnet worden ist, sollte eine Fehlerbehandlung dafür Sorge tragen, dass – falls sich die Vorlage nicht im vordefinierten Verzeichnis befindet – das Makro zunächst abgebrochen wird.
9. Speichern und testen Sie Ihre Anwendung.
10. Sie werden beim Testen feststellen, dass die Anfrage des Kalenderjahres nicht sofort sichtbar ist, da EXCEL standardmäßig nicht aktiviert ist. Für den Ablauf ist es noch besser, wenn Sie vor dem Abruf der Jahresplanvorlage EXCEL aktivieren:
AppActivate "Microsoft Excel"
11. Und am Ende wieder Access:
AppActivate "Microsoft Access"



Sie haben Ihren Jahresplan von ACCESS aus gestartet. Wie immer finden Sie die Komplettlösung im Anhang und in der entsprechenden Beispieldatei (SEMI_03.MDB).

4.2.3 Datensätze einer Datenbank lesen

Da der Jahresplan für die Raumplanung jetzt die Information benötigt, wann welches Seminar im ausgewählten Raum geplant ist, muss nun ein Zugriff auf die Datenbank realisiert werden.

VBA bietet dazu zwei Datenzugriffsmethoden an:

DAO (Data Access Objects)

Den Zugriff über DAO gab es in allen bisherigen Versionen und wird auch weiterhin unterstützt, jedoch nicht weiterentwickelt. Methoden wie **OpenDatabase** und **OpenRecordset** ermöglichen den Zugriff mit dieser Variante. Für den Zugriff über DAO benötigen Sie einen ODBC-Treiber.

ADO (ActiveX Data Objects)

Diese Zugriffstechnik wird erst seit der Version VBA 2000 angeboten und löst die bisherige DAO-Zugriffsmethode ab. Der Zugriff erfolgt über OLE DB. Damit ist es nicht nur möglich, auf relationale Datenbanken mittels ODBC zuzugreifen. Auch der Zugriff von ASCII-, WORD und z.B. VSAM – um nur einige zu nennen – ist mit OLE DB möglich.

**Neu in
VBA 2000:
ActiveX Data
Objects für
den Zugriff
auf Daten**

Bevor Sie jedoch die Befehle eingeben können, sollten Sie sich über die benötigte Abfrage im Klaren sein. Gebraucht werden alle Termine des ausgewählten Raums. Am besten definieren Sie zunächst eine Abfrage, aus der Sie dann die SQL-Syntax entnehmen können, um sie in Ihrem Programm zu implementieren.

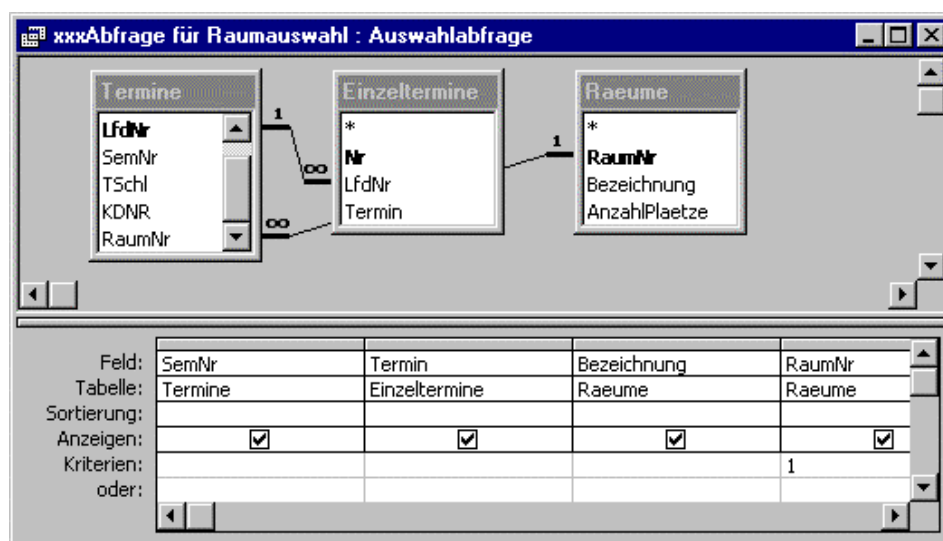


Abb. 4.5: Abfrage für Raumauswahl



Wenn Sie jetzt die SQL-Ansicht mit Hilfe des Befehls Ansicht → SQL-Ansicht aktivieren, können Sie die benötigte SQL-Anweisung in die Zwischenablage kopieren.

```

SELECT Termine.SemNr, Einzeltermine.Termin, Raeume.Bezeichnung, Raeume.RaumNr
FROM Raeume INNER JOIN (Termine INNER JOIN Einzeltermine ON Termine.LfdNr =
Einzeltermine.LfdNr) ON Raeume.RaumNr = Termine.RaumNr
WHERE (((Raeume.RaumNr)=1));

```

Abb. 4.6: SQL-Ansicht der Abfrage

Beachten Sie bitte auch, dass die fest definierte Bedingung **RaumNr=1** später durch die ausgewählte Raumnummer des Formulars ersetzt werden muss.

ADO Zugriff realisieren

Sie realisieren den Zugriff auf die Daten der Datenbank mit einer SQL-Abfrage.



1. Da der Ablauf nicht sinnvoll ist, falls kein Seminar ausgewählt wurde, sollte dieser Fall zunächst abgefangen werden. Die Anweisungen können Sie direkt nach dem Klicken der Schaltfläche, noch vor dem Start von EXCEL, realisieren. Mit der Funktion **IsNull** können Sie prüfen, ob eine Auswahl erfolgt ist:

```

If IsNull(Me.Raumauswahl) Then
    MsgBox "Bitte erst Raum auswählen"
    Me.Raumauswahl.SetFocus
    Exit Sub
End If

```

Das Objekt **me** ist in diesem Fall das aktuelle Formular. Die Angabe ist für die Realisierung nicht unbedingt erforderlich, sie hilft jedoch bei der Erfassung, weil entsprechende Hilfenfenster geöffnet werden.

2. Deklarieren Sie eine lokale Variable vom Typ String:

```
Dim strSQL As String
```

3. Weisen Sie der Stringvariablen den kopierten SQL-String zu:

```
strSQL =  +  (Inhalt der Zwischenablage einfügen)
```

```

cmdExcel Click
AppActivate "Microsoft Access"
strsql = SELECT Termine.SemNr, Einzeltermine.Termin
FROM Raeume INNER JOIN (Termine INNER JOIN Einzeltermine
WHERE ((Raeume.RaumNr)=1));
End Sub

```

Abb. 4.7: SQL-Abfrage aus der Zwischenablage einfügen



4. Bilden Sie einen gültigen String, indem Sie mit der Stringverkettung (Verkettungszeichen: &) die Textfolgen aneinanderreihen. Berücksichtigen Sie dabei auch, dass die Konstante 1 durch die eigentliche Raumauswahl aus dem Formular ersetzt werden muss.

```

End If
On Error GoTo 0
AppActivate "Microsoft Access"
strSQL = "SELECT Termine.SemNr, Einzeltermine.Termin, " & _
    "Raeume.Bezeichnung, Raeume.RaumNr FROM Raeume " & _
    "INNER JOIN (Termine INNER JOIN Einzeltermine ON " & _
    "Termine.LfdNr = Einzeltermine.LfdNr) ON Raeume.RaumNr " & _
    "= Termine.RaumNr WHERE ((Raeume.RaumNr)=" & _
    & Me.Raumauswahl & ");"
End Sub

```

Abb. 4.8: SQL-Text überarbeiten

5. Für den ADO-Zugriff benötigen Sie noch zwei Objektvariablen vom Typ **ADODB.Recordset** und **Connection**.
Dim RS As ADODB.Recordset, CON As ADODB.Connection
 Sie definieren sie selbstverständlich am Prozeduranfang.
6. Mit dem **Connection**-Objekt bestimmen Sie die Datenbank. Wenn es sich um die aktuelle **Datenbank** handelt, kann die Zuweisung wie folgt aussehen:
Set CON = CurrentProject.Connection
7. Das **Recordset**-Objekt bestimmt die Tabelle oder Abfrage. Da Sie eine neue Abfrage mit einem SQL-String definieren wollen, kann die Zuweisung ein Verweis zu einem neuen **Recordset**-Objekt sein:
Set RS = New ADODB.Recordset
 Alternativ hätten Sie die Deklaration auch direkt als neue Instanz der Klasse **Recordset** wie folgt deklarieren können:
Dim RS As New ADODB.Recordset
8. Mit der **CursorType**-Eigenschaft bestimmen Sie noch die Art des Zugriffs. Da unsere Abfrage nur sequentiell vorwärts abgearbeitet wird, sollte die Zuweisung wie folgt aussehen:
RS.CursorType = adOpenForwardOnly
 Mehr Informationen zu dieser Eigenschaft sind in der Online-Hilfe abrufbar.



9. Jetzt kann die Abfrage mit der **Open**-Methode geöffnet werden:
RS.Open strSQL, CON,,, adCmdText
Beachten Sie den Bezug zur SQL-Abfrage (String strSQL) und die Verbindung zur Datenbank (CON). Die Konstante adCmdText gibt an, dass der Provider (in unserem Fall ACCESS) den Parameter **source** als Textdefinition eines Befehls auswertet.
Mehr zum Thema Provider erfahren Sie in der Online-Hilfe unter dem Kapitel: „Verwenden von Providern mit ADO“.

10. Mit einer Schleife (bis Abfragenende=Eigenschaft **EOF**) können Sie jetzt die einzelnen Datensätze bearbeiten. Um endlich wieder einmal ein Zwischenergebnis zu sehen, lassen Sie sich die einzelnen Termine mit einer **MsgBox**-Anweisung anzeigen.
Do While Not RS.EOF
MsgBox RS.Fields("Termin")
RS.MoveNext
Loop
Mit Hilfe der **Fields**-Auflistung können Sie Bezug auf die einzelnen Felder nehmen, um dann mit der Methode **MoveNext** eine Positionierung zum nächsten Datensatz vorzunehmen.

11. Schließen Sie zum Abschluss zuerst das **Recordset**- und anschließend das **Connection**-Objekt mit den entsprechenden **Close**-Methoden:
RS.Close
CON.Close

12. Speichern und testen Sie Ihre Anwendung.

Die Komplettlösung finden Sie am Ende der nächsten Übung.

Betrachten Sie bitte die beiden nachfolgenden Abbildungen. Sie sollen helfen, den letzten Teil der Aufgabe besser zu verstehen.

Seminar-Nr	Termin	Bezeichnung	RaumNr
EXE	14.06.1999	Berlin	1
EXE	15.06.1999	Berlin	1
WOE	16.06.1999	Berlin	1
WOE	17.06.1999	Berlin	1
PPE	21.06.1999	Berlin	1
WOE	28.06.1999	Berlin	1
WOE	29.06.1999	Berlin	1
VBA	02.08.1999	Berlin	1
VBA	03.08.1999	Berlin	1
VBA	04.08.1999	Berlin	1
PPE	15.06.1999	Berlin	1
PPE	16.06.1999	Berlin	1

Abb. 4.9: Beispielergebnis einer Raumplanabfrage



	A	C	D	E	F	G	H
1	1999	Februar	März	April	Mai	Juni	Juli
12	11			So			So
13	12					Sa	
14	13	Sa	Sa		Chr.Hi.	So	
15	14	So	So		----	exe	
16	15	Karneval			Sa	exe/ppe	
17	16				So	woe/ppe	
18	17			Sa		woe	Sa
19	18			So			So

Abb. 4.10: Beispielergebnis fertiger Plan

Sie übertragen die geplanten Seminare der Datenbank in Ihre Excel-Tabelle gemäß dem abgebildeten Beispielplan.

1. Für die Übertragung der Seminartermine in den Excel-Jahresplan benötigen Sie zunächst ein **Range**-Objekt.

Dim Zelle As Range

2. Im **Range**-Objekt speichern Sie den Bezug zur Zelle, die in EXCEL mit den Seminareinträgen ergänzt werden soll. Dabei hilft Ihnen wieder der bereits in EXCEL definierte Name **Jahresplan**. Die Information Tag und Monat des Seminartermins verweist wieder auf den relativen Zeilen- und Spaltenindex des Jahresplans.

```
Set Zelle = oApp.Range("Jahresplan").Cells(_
    Day(RS.Fields("Termin")), _
    Month(RS.Fields("Termin")))
```

3. Jetzt kann mit dem Objekt **Zelle** direkt Bezug genommen werden. Da auch mehrere Einträge zu einer Zelle kommen können, prüfen Sie zunächst, ob die Zelle bereits einen Inhalt aufweist. Wenn ja, ergänzen Sie den aktuellen Eintrag um den neuen Wert, wobei der Schrägstrich als Trennzeichen dient. Sollte die Zelle leer sein, kann das Seminarkürzel direkt eingetragen werden:

```
If Zelle <> "" Then
    Zelle = Zelle & "/" & LCase(RS.Fields("SemNr"))
Else
    Zelle = LCase(RS.Fields("SemNr"))
End If
```

Die Funktion **LCase** konvertiert das Seminarkürzel in Kleinbuchstaben. Damit steht mehr Platz in der Zelle zur Verfügung.

Ergänzung Excel- Jahresplan





4. Bisher überprüfen Sie noch nicht, ob die ankommenden Daten überhaupt im geforderten Zeitraum liegen. Eine Lösung wäre, die **where**-Klausel der SQL-Abfrage um eine weitere Bedingung zu ergänzen. Alternativ dazu könnte die Schleife zur Bearbeitung der Sätze diese Prüfung vornehmen. Der nachfolgende Programmcode zeigt eine Lösung zur zweiten Möglichkeit:

```
Dim intKalJahr As Integer
...
intKalJahr = oApp.Range("Kalenderjahr")
...
If Year(RS.Fields("Termin")) = intKalJahr Then
...
End If
```

Die Zeichen ... verweisen auf bereits fertigen Programmcode. Die Positionen ergeben sich aus der Programmlogik.

5. Es wäre schön, wenn aus der Excel-Tabelle zu erkennen wäre, für welchen Raum die Planung vorgenommen worden ist. Erreichen können Sie dies, indem Sie den Blattnamen der Excel-Tabelle entsprechend verändern:

```
oApp.Sheets(1).Name = "Raumplanung " & RS.Fields("Bezeichnung")
```

6. Sie könnten dem Anwender mitteilen, dass der Programmablauf fertig ist. Sinnvoll wäre die gleichzeitige Frage, ob die Positionierung nach ACCESS oder nach EXCEL erfolgen soll. Beachten Sie dazu folgende Befehle:

```
AppActivate "Microsoft Access"
If MsgBox("Fertig! Jahresplan aktivieren?", vbYesNo + vbQuestion) _
= vbYes Then
AppActivate "Microsoft Excel"
End If
```

7. Speichern und testen Sie Ihre Anwendung.

Eine gesamte Auflistung der fertigen Ereignisprozedur finden Sie im Anhang Lösungen und in der Beispieldatenbank SEMI_04.MDB.

Zusammenfassung

Bedienung von EXCEL von ACCESS

Die Bedienung einer anderen Anwendung in VBA ist jederzeit möglich. Voraussetzungen: die andere Anwendung muss installiert und die Bibliothek mit den benötigten Befehlen der Anwendung bekannt gemacht sein. Zur Verwendung der Befehle benötigt man dann eine Objektvariable mit einem Verweis zur entsprechenden Anwendung. Diese Objektvariable muss dem eigentlichen Befehl quasi als Objektvorsatz mitgegeben werden.

Datenzugriff

Wenn Sie auf die Daten einer Datenbank zugreifen wollen, können Sie die Abfrage in ACCESS entwickeln und den SQL-Text über die Zwischenablage in Ihren Programmcode einfügen. So sind über Textverkettung auch umfangreiche variable Abfragen jederzeit realisierbar. Sie benötigen dazu ein **Connection**- und **Recordset**-Objekt aus der ADO-DB-Zugriffsbibliothek.



Testaufgaben

- ▶ Frage 1: Die Buchstabenfolge ADO steht für folgende Wörter:
 - A. ActiveX Data Object.
 - B. Application Data Object.
 - C. Application Direct Organization.

- ▶ Frage 2: Um Befehle einer anderen Anwendung zu verwenden, treffen folgende Aussagen zu:
 - A. Die andere Anwendung muss lediglich installiert sein.
 - B. Die Anwendung muss nicht installiert sein, es reicht ein Verweis zur entsprechenden Bibliothek.
 - C. Die andere Anwendung muss installiert sein und es muss ein Verweis zur entsprechenden Bibliothek vorgenommen werden.

- ▶ Frage 3: Mit den Funktionen **GetObject** und **CreateObject** können folgende Aktionen durchgeführt werden:
 - A. Man kann eigene Objekte damit definieren.
 - B. Man kann einen Verweis zu einer Applikation erstellen und sie bei Bedarf öffnen.

- ▶ Frage 4: Welche Aussagen zum Thema Datenzugriff treffen zu:
 - A. Einen Datenzugriff auf eine Datenbank kann mit DAO-Objekten realisiert werden.
 - B. Der Zugriff über ADO-Objekte ist die neuere, moderne Variante des Datenzugriffs.
 - C. Damit man auf eine Tabelle von ACCESS zugreifen kann, muss auf jeden Fall ACCESS installiert sein.
 - D. OLE DB ist keine spanische Veranstaltung, sondern löst den bekannten Zugriff auf Daten über ODBC aus. Es werden noch mehr verschiedene Datenquellen unterstützt. Der Zugriff in der Programmierung erfolgt über ADO-Objekte.



5 Access, Word und VBA

Ziele dieses Kapitels

- ▶ Sie lernen das Zusammenspiel von Ereignisprozeduren in ACCESS.
- ▶ Sie schreiben Ihr erstes Makro in WORD.
- ▶ Sie erstellen ein benutzerdefiniertes Formular in WORD.
- ▶ Sie stellen erneut eine Verbindung zwischen Office-Produkten her, indem Sie Ihre Word-Vorlagen in ACCESS ausfüllen.

5.1 Ereignisprozeduren in Access

In keinem anderen Office-Programm werden Sie so intensiv mit Ereignisprozeduren arbeiten wie in ACCESS. Sie bestimmen die Grade der Benutzerfreundlichkeit und der Sicherheit Ihrer Anwendung.

Dieses Kapitel zeigt Ihnen das Zusammenspiel der Komponenten Ereignisse, Eigenschaften und Methoden am Beispiel von Seminaranmeldungen. Der Arbeitsplatz Seminarverwaltung hat u.a. die Aufgabe, Seminaranmeldungen einzugeben und dann die Bestätigungen zu verschicken.

Dazu müssen folgende Aktionen ablaufen, deren Automatisierung Aufgabe dieses Kapitels ist.

- ▶ Das Formular Anmeldung muss in ACCESS aufgerufen werden.
- ▶ Nachdem das Seminar und der Termin ausgewählt wurden, müssen die bereits erfassten Anmeldungen sichtbar werden.
- ▶ Jetzt kann die neue Anmeldung gebucht werden. Falls der Teilnehmer noch nicht gespeichert ist, muss es eine Möglichkeit geben, das Adressformular direkt von der Anmeldung aufzurufen.
- ▶ Nachdem die Anmeldung gespeichert ist, sollte die Bestätigung automatisch ausgedruckt werden, und zwar unter Verwendung einer Word-Vorlage.

5.1.1 Vorbereitung

Für die Realisierung benötigen Sie einige Access-Formulare, die wir Ihnen nachfolgend kurz vorstellen. Falls Sie über die Beispieldaten verfügen, können Sie sich die Vorbereitung sparen. Sie finden sie in der Datenbank SEMI_10.MDB.